TCP Throughput and Buffer Management

Todd Lizambri, Fernando Duran, and Shukri Wakid
National Institute of Standards and Technology
Building 222, Room A353, 100 Bureau Drive
Gaithersburg, MD 20899-3207, U.S.A.
(301) 975-8074 (Voice), (301) 975-8254 (FAX)
E-mail: {todd.lizambri, fernando.duran, shukri.wakid}@nist.gov

Abstract

There have been many debates about the feasibility of providing guaranteed Quality of Service (QoS) when network traffic travels beyond the enterprise domain and into the vast unknown of the Internet. Many mechanisms have been proposed to bring QoS to TCP/IP and the Internet (RSVP, DiffServ, 802.1p). However, until these techniques and the equipment to support them become ubiquitous, most enterprises will rely on local prioritization of the traffic to obtain the best performance for mission critical and time sensitive applications. This work explores prioritizing critical TCP/IP traffic using a multi-queue buffer management strategy that becomes biased against random low priority flows and remains biased while congestion exists in the network. This biasing implies a degree of unfairness but proves to be more advantageous to the overall throughput of the network than strategies that attempt to be fair. Only two classes of services are considered where TCP connections are assigned to these classes and mapped to two underlying queues with round robin scheduling and shared memory. In addition to improving the throughput, cell losses are minimized for the class of service (queue) with the higher priority.

1. Introduction

The most effective way to design Internet routers is to understand the performance behavior of TCP flows, thus enabling the management of multiple parallel connections that share common resources such as memory. Such connection management can be provided via a peer level (or even an application like) software that interfaces with TCP services or it can be provided via intelligence in the underlying buffer management scheme.

As has been obvious through numerous publications by the Internet community, the TCP performance behavior has two aspects to it; protocol design and implementation. Inherent protocol design mechanisms such as flow control with the slow window start, the FIFO reception of the acknowledgements and related delay, the connection management and retransmissions during packet losses have been well known problems. The impact of implementation aspects on TCP performance, however, can be even more significant. Packet discard policies, buffer management and scheduling, congestion control, and packet processing time have been shown to make a significant impact on performance [1,2,3].

In principle, one would like to be able to manage the various TCP connections independently while honoring some quality of service for each. However, this is not only a very complex problem, but it can hardly be practical considering current operating system designs and typical machine/network resources. To overcome these problems, we propose a buffer management scheme that provides maximum throughput for the high priority traffic while eliminating exponential degradation of low priority traffic. We consider only two types of TCP services, one of which has a priority edge and is designed for greater effective throughput by minimizing packet loss and retransmission of data. In this context, we map these two types to two logically separate underlying queues and strive to maximize the throughput of TCP. Since we are studying the behavior of the end systems, we only concentrate on ABR and UBR like services rather than emphasizing the various possible ATM qualities of service [4]. This paper focuses only on the effects of end system buffering and scheduling of TCP/IP traffic over ATM networks and does not consider other related system effects, such as, overall network congestion and switch and destination flow control for ABR services.

This paper has been organized as follows: Section 2 describes the simulation environment that was used for our experimentation. In Section 3, we describe the admission control and cell discard policy used in the traffic shaper. Section 4 presents the results for the admission control policy and how the throughputs of the high and low priority queues are affected. Section 5 provides our conclusions.

2. Simulation

We simulate TCP/IP over an ATM configuration where 10 active 60 Mb/s connections are competing over an OC-3[1] physical medium (see Figure 1). This stressful condition allows us to exacerbate the known TCP performance problems and examine those policies that would permit a steady state throughput even under such conditions. TCP throughput is defined here as the number of *acknowledged* bytes per unit time. Retransmissions due to non-delivered packets are not counted in this computation. Each simulation was run for a 10-second duration and consisted of 10 sending applications with an infinite amount of data to send and 10 receiving applications that did not send data (except for acknowledgements). The connection start times were staggered at different intervals with times of 0 ms, 0.05 ms, and 0.5 ms respectively. We also assume a receiver window size of 32 K bytes and a maximum MAC layer segment of 1460 bytes (Ethernet payload size minus TCP/IP headers). A MAC layer shared buffer capacity of 3000 cells is used. These assumptions are realistic and are based on our experience with running and validating the ATM simulator [2].

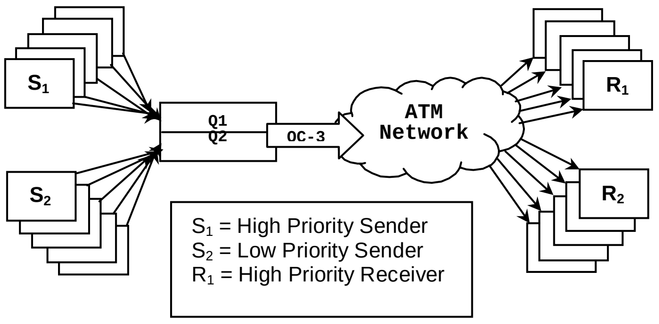
S_1
R_1
Q1
Q2
OC-3
ATM Network
S_2
R_2
S_1 = High Priority Sender
S_2 = Low Priority Sender
R_1 = High Priority Receiver

Figure 1. Simulation configuration

3. Admission Control Policy

The two queues (Q1, Q2) use shared memory where at some high utilization (say 80 %), the high priority queue (Q1) is ONLY allowed "ownership" of the available memory space [5]. At this high utilization threshold, Q2 will deactivate one of its connections and keep track of its identification (ID). As the available memory space decreases beyond the threshold in certain increments of time, Q2 will deactivate more of its connections (for other thresholds) and keep track of their identifications. This of course can go on until Q2 deactivates all of its connections in a worst case scenario. However, if the available memory lingers around the first threshold (80%), the first connection that was deactivated will get a high activation priority through a fairness mechanism. This is called *flow biasing.* All flows/connections in Q1 are biased against those of Q2. However, within Q2, the flows that are deactivated first will get an activation edge when available memory is above the configured threshold. This policy, of course, guarantees a better quality of service for Q1. However, to optimize TCP throughput, we also invoke a packet discard policy for Q2. When one cell is discarded in a connection that was deactivated in Q2, the remaining cells for the TCP packet, for which such cell belongs to, are also discarded. This strategy, know as *Partial Packet Discard* (PPD), is looked at in detail in [6]. The pseudo code for the admission control and flow biasing can be found in Figure 2.

```
FOR EACH INCOMING CELL
   IF CELL SERVICE CLASS = LOW PRIORITY THEN
      IF AVAILABLE_MEMORY < BIAS_THRESHOLD[FLOW] THEN
          -DISCARD CELL
          -ENTER PARTIAL PACKET DISCARD MODE FOR COMPLETE PACKET
          IF FLOW IS ACTIVE THEN
             -DEACTIVATE FLOW: DROPPING  CELLS FOR FLOW
              UNTIL AVAILABLE_MEMORY > BIAS_THRESHOLD[FLOW]
          ELSE FLOW IS ALREADY DEACTIVATED
             -DECREASE BIAS_THRESHOLD[FLOW] TO INCREASE
              PROBABILITY OF REACTIVATION
          END IF
          ELSE AVAILABLE_MEMORY > BIAS_THRESHOLD[FLOW] THEN
             -RE-ACTIVATE FLOW: ACCEPT CELLS STARTING WITH NEXT
              COMPLETE PACKET
          END IF
   ELSE IF SERVICE CLASS = HIGH PRIORITY THEN
      IF AVAILABLE_MEMORY = 0 & LOW PRIORITY CELLS EXIST
          -REMOVE LOW PRIORITY CELL FROM MEMORY
          -DEACTIVATE FLOW: DROPPING ALL CELLS FOR FLOW UNTIL
           AVAILABLE_MEMORY > BIAS_THRESHOLD[FLOW]
          -ACCEPT THE HIGH PRIORITY CELL
      ELSE IF AVAILABLE_MEMORY = 0 & LOW PRIORITY CELLS DO NOT
EXIST
          -DISCARD HIGH PRIORITY CELL
          -ENTER PARTIAL PACKET DISCARD MODE FOR COMPLETE PACKET
      ELSE AVAILABLE_MEMORY > 0
          -ACCEPT CELL
      END IF
   END IF
END FOR
```

Figure 2. Admission control and flow biasing pseudo code

4. Results

We extended a NIST ATM simulator [7] to incorporate buffer management techniques, traffic shaping algorithms, and TCP/IP protocols. The significant use of this simulator by the international ATM community and the comprehensive testing & debugging that we performed on its expanded features make us comfortable with its validity.

We use flow invocation sequences shown in Table 1 for 10 connections, 5 of which are assigned to each queue.

[1] Optical Carrier 3 at 155 Mb/s

Table 1. Total TCP throughput and cell loss

Start Time Δt (ms)	Buffer Management Policy	Q1 Throughput (Mb/s)	Q2 Throughput (Mb/s)	Total Throughput (Mb/s)	Cell Loss Q1/Q2
0	No Flow Bias	75.92	30.62	106.54	43/4211
	Flow Bias 80%	90.96	10.04	101.00	0/6800
	Flow Bias 90%	90.96	10.09	101.05	0/6555
	Flow Bias 95%	90.96	21.52	112.48	0/5520
0.05	No Flow Bias	83.52	30.20	113.72	46/4438
	Flow Bias 80%	90.99	13.26	104.25	0/6962
	Flow Bias 90%	90.96	16.70	107.66	0/6497
	Flow Bias 95%	91.07	30.15	121.22	0/4578
5	No Flow Bias	83.49	22.47	105.96	50/5163
	Flow Bias 80%	90.81	12.79	103.60	0/6142
	Flow Bias 90%	90.81	11.22	102.03	0/4637
	Flow Bias 95%	90.70	11.61	102.31	0/6451

The "Start Time" left column of this table provides the various units of time at which flow invocations were triggered. For time intervals of 0 ms, all the 10 flows were invoked at the same time, while for time interval of 0.05 ms, each flow started 0.05 ms after the previous one. The "Buffer Management Policy" column uses various profiles of Partial Packet Delays (PDD) together with different buffer availability thresholds to invoke flow biasing. Then the corresponding Q1, Q2 and total throughputs are provided with the related cell losses in the remaining columns. Out of a numerous set of experiments, we selected those thresholds and attributes that provide a better quality of service and optimal TCP throughput. At time interval of 0 ms, where all the connections are simultaneously active, the tail-dropping packet strategy combined with the simultaneous burst of data causes synchronization of TCP windows thus leading to a slightly poorer performance than that obtained by 0.05 ms and 0.5 ms intervals. It is important to note that during this simulation we used a mean packet-processing overhead of 300 µs (with a standard deviation of 15 µs) [1,3]. This is a realistic value we have obtained using reliable measurements at NIST and considering the limitations of the simulator that would not allow different processing times for sending, receiving, and acknowledging data [1]. Since TCP throughput is quite sensitive to such an overhead [3], the data in Table 1 is quite realistic and shows an average greater than 10 Mb/s throughput per connection. As shown in [1], the throughput bound due to processing time can be calculated as the amount of data divided by the processing time elements, but ignoring transmission time.

$$\begin{aligned} \textit{Max Throughput} &= \frac{\textit{Amount of Data Sent}}{\textit{Processing Delays}} \\ &= \frac{1460\ \textit{bytes/packet} \times 8\ \textit{bits/byte}}{\sim 600\times10^{-6}\ \textit{seconds / packet}} \\ &= \sim 19.5\ \textit{Mb/s} \end{aligned}$$

For our simulation, the combined processing time is approximately 600 µs, which yields a maximum throughput of 19.5 Mb/s per connection. Since each queue is fed by 5 TCP connections, the maximum throughput per queue is approximately 97.5 Mb/s (the deviation in processing time may cause this number to vary slightly). Berkley's System V Release 4.0 with Reno was used for the TCP/IP stack. Selective acknowledgement logic (SACK) was not used. For all simulation runs, the receiver window size was 32K bytes and the maximum segment size was 1460 bytes (sized for Ethernet frames). The amount of memory allocated to the queuing was 3000 cells. This memory was shared between Q1 and Q2 based on the flow biasing algorithm. The output link of the traffic shaper was fixed at 155 Mb/s.

Figures 3 to 6 display the effective throughput for Q1 and Q2 with varying values of flow biasing from the simulation runs that have a staggered flow start time of 0.05 ms (see row 2 of Table 1). Figure 3 shows a baseline without the flow biasing. In this figure, both queues incur cell loss and Q1 does not achieve maximum throughput. The variance of throughput in Q2 (including periods of no transmission) occur when cells are lost and a flow halts transmission for the duration of the TCP retransmission timeout. One can visually determine the number of flows *biased* (or halting transmission) during a given period of time. With flow biasing (Figures 4 to 6), maximum throughput is achieved in Q1. The varying threshold of memory usage to determine when biasing should occur affects the throughput of the Q2.

Our goal was to eliminate the cell loss and achieve maximum throughout in the high priority queue (Q1) while maximizing the throughput in Q2. The constant steady TCP throughput of about a 91 Mb/s in Q1 under total input stress conditions of 600 Mb/s while maintaining a throughput of 30.15 Mb/s in Q2 simply shows the value of the above policy when biasing occurs at 95% memory utilization.

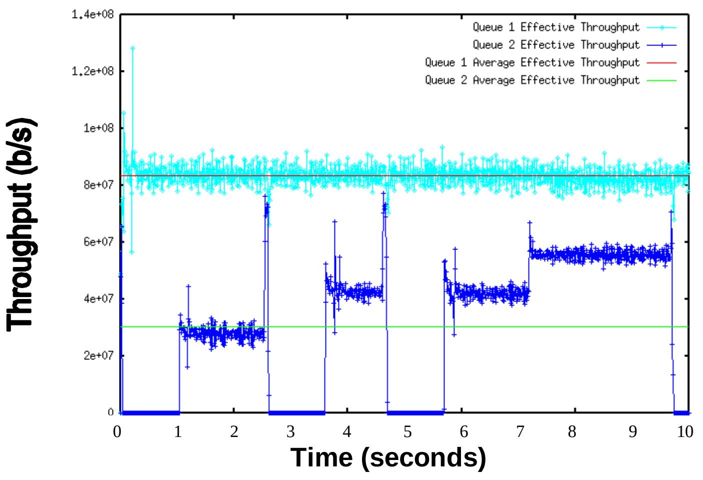

Figure 3. TCP Effective Throughput vs. Time with Shared Buffer Space & No Flow Biasing

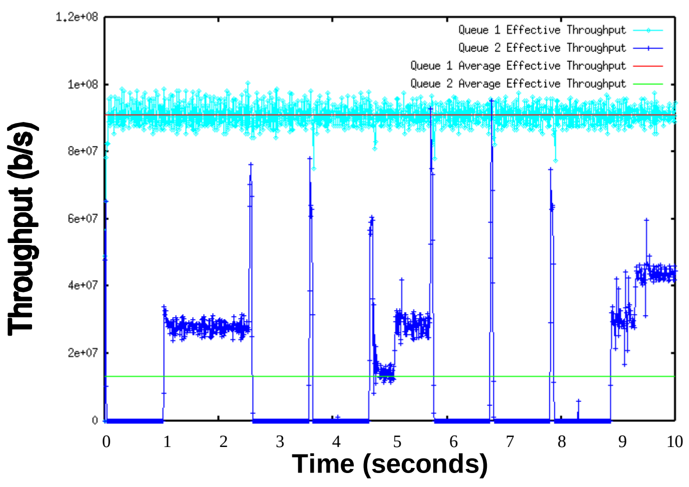

Figure 4. TCP Effective Throughput vs. Time with Flow Biasing when Shared Buffer is 80% Full

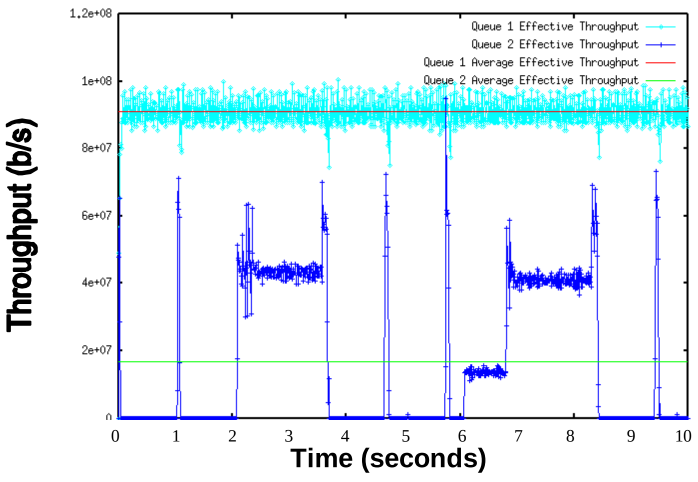

Figure 5. TCP Effective Throughput vs. Time with Flow Biasing when Shared Buffer is 90% Full

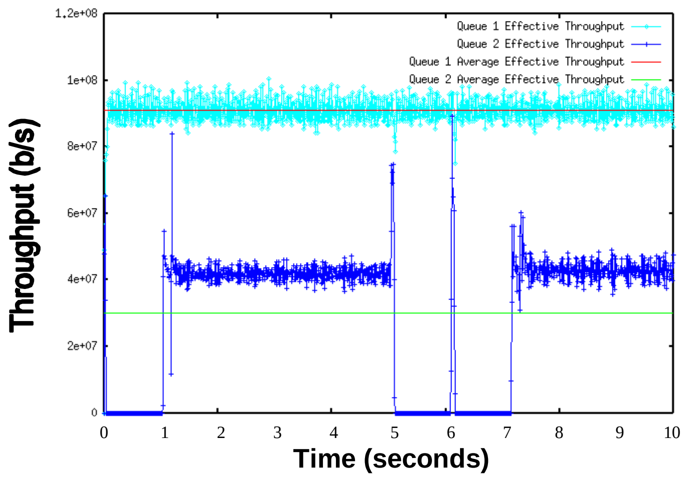

Figure 6. TCP Effective Throughput vs. Time with Flow Biasing when Shared Buffer is 95% Full

5. Conclusions

We presented here a policy that prevents TCP from the known exponential degradation of throughput under stressful conditions. Such policy enables a steady overall two-thirds effective utilization of the ideal and inherent available bandwidth in the physical medium, while achieving maximum theoretical throughput in the high-priority queue. We also, and in parallel, provide a means for minimizing packet loss for a subset class of services.

References

[1] Y. Fouquet, A. Mink, and S. Wakid, “Hardware Measurement Techniques for High Speed Networks”, Journal of High Speed Networks, vol. 3, no. 2, p 187, 1994.

[2] T. Lizambri, F. Duran, and S. Wakid, “Priority Scheduling and Buffer Management for ATM Traffic Shaping”, Proceedings of the 7th IEEE Workshop on Future Trends of Distributed Computing Systems, FTDCS 99.

[3] D. Clark, V. Jacobson, J. Romkey, and H. Salwen, “An Analysis of TCP Processing Overhead”, IEEE Communications Magazine, June 1989.

[4] ATM Forum, "ATM Traffic Management Specification Version 4.0" April 1996, ftp://ftp.atmforum.com/pub/aproved-specs/af-tm-0056.000.ps

[5] G.Wu, J.Mark, “A Buffer Allocation Scheme for ATM Networks: Complete Sharing Based on Virtual Partition”, IEEE/ACM Transactions on Networking Vol. 3, Num. 6, Dec 95.

[6] A. Romanow and S. Floyd, “Dynamics of TCP Traffic over ATM Networks”, IEEE Journal on Selected Areas in Communications, vol. 13, no. 4 (May 1995), 633-641.

[7] NIST ATM simulator, http://www.hsnt.nist.gov/misc/hsnt/prd_atm-sim.html

Acronyms

ABR:	Available Bit Rate
ATM:	Asynchronous Transfer Mode
FIFO:	First In First Out queue
IP:	Internet Protocol
MAC:	Medium Access Control
NIST:	National Institute of Standards & Technology
OC-3:	Optical Carrier-3
PDD:	Partial Packet Discard
RSVP:	ReSerVation Protocol
TCP:	Transmission Control Protocol
UBR:	Unspecified Bit Rate